Amendments to the Specification:

Please replace the paragraph starting at page 2, line 25 with the following amended paragraph:

Three variables that determine practicality to the end user are portability, affordability, and value. Fat client devices, while benefiting from additional functionality, usually suffer a decrease in portability, affordability, product practicality, and mainstream adoption. In addition, a closer look at the functionality actually being delivered by such fat client devices reveals further limitations. For example, although such devices can usually access simple POP3 and IMAP4 email accounts, they may not be sophisticated enough to negotiate corporate firewalls or communicate with proprietary servers (e.g., ~~Microsoft Exchange~~ MICROSOFT EXCHANGE™ and ~~Lotus Domino~~ LOTUS DOMINO™) to access email or PIM data. As a result, corporate end users must maintain separate email accounts for their wireless HCDs and will have no access to corporate server-based PIM data.

Please replace the paragraph starting at page 3, line 20 with the following amended paragraph:

With the wide-ranging proliferation of the Internet, so-called "web-based applications" have become highly prevalent. Popular sites (some examples may be ~~Hotmail, Yahoo! Mail, Yahoo! Calendar~~ HOTMAIL™, YAHOO! MAIL™, YAHOO! CALENDAR™, and ~~Microsoft Investor~~ MICROSOFT INVESTOR™) provide users with a web interface to the kinds of applications that were previously only available as client side software. At one level, the term "application" seems accurate, but the usage model of a classic client-side application and a web-based application differ considerably. In contrast to the client-side model, web-based applications are stateless and non-interactive. For example, every click of the end user's mouse, selection on a menu, or

update requires a reconnection to the server and a refresh of the web page. Even over the fastest Internet connections the user experience on a web-based application is arduous when compared to the persistent, interactive nature of client-side applications. Another drawback of this approach is that web-based email applications require their users to manage yet another email address. These approaches cannot function in the true sense of a desktop application, i.e., as a tool to reach individual source data instead of a service.

Please replace the paragraph starting at page 4, line 14 with the following amended paragraph:

As the Internet started gaining momentum and the static and stateless nature of web pages became apparent, new technologies such as ~~Java, ActiveX,~~ JAVA™, ACTIVEX™, and dynamic hypertext markup language (DHTML) were developed. The growing popularity of wireless HCDs and the inadequacies of the static web view will again prompt competition related to the next development platform in the wireless market.

Please replace the paragraph starting at page 5, line 5 with the following amended paragraph:

An initial response to ~~Java~~ JAVA™ was ~~ActiveX~~ ACTIVEX™, and while that solution is very effective in certain scenarios, the lack of platform independence may prove to be its downfall. A recent response to ~~Java~~ JAVA™ is DHTML, which incorporates client-side scripting in conjunction with HTML to provide a user experience that is far more interactive than plain HTML while retaining platform independence. However, at one level, DHTML is very similar in concept to a virtual machine. Rather than having an actual virtual machine, DHTML uses scripts and snippets of code in much the same way a ~~Java~~ JAVA™ virtual machine does. In this regard, the browser

functions as a layer between the application and the OS, and therefore suffers from many of the same limitations as a virtual machine.

Please replace the paragraph starting at page 5, line 15 with the following amended paragraph:

Unlike most of the so-called "thin client" technologies discussed herein, ~~ActiveX~~ ACTIVEX™ leverages the OS and platform directly, making it a powerful solution for "web-accessed" (as opposed to "web-based") applications. However, because of this, ~~ActiveX~~ ACTIVEX™ is OS-dependent and processor-dependent, making it a poor solution for the HCD space where multiple OS and processor configurations abound. Furthermore, ~~ActiveX~~ ACTIVEX™ is in some ways a return to the fat client concept of installing client-side software for local processing.

Please replace the paragraph starting at page 5, line 22 with the following amended paragraph:

With the increase in network bandwidth, one of the oldest client-server architectures is making a resurgence. Solutions such as ~~Citrix, X-Windows, Windows Terminal Server,~~ CITRIX™, X-WINDOWS™, WINDOWS TERMINAL SERVER™, and ~~PC Anywhere~~ PC ANYWHERE™ are growing in popularity as corporate IS professionals scramble to lower total cost of ownership. All of these solutions employ a thin client that can be ported to multiple platforms, and provide the user with a full graphical representation of their applications running on a remote server.

Please replace the paragraph starting at page 5, line 28 with the following amended paragraph:

By using this type of arrangement, corporations may employ a system where all of their users access applications from a single ~~Windows 2000~~ WINDOWS 2000™ server through simple clients (such as ~~Windows CE~~ WINDOWS CE™ based terminals) located on their desktops. The advantage to the corporation is that this system allows multiple users to share resources with a single point of administration, making the entire system easier to support. The downside is that it also presents a centralized point of failure.

Please replace the paragraph starting at page 7, line 11 with the following amended paragraph:

The result is that there is no need to "round-trip" every keystroke, since such inputs can be produced using client-side controls. Data can then be transmitted in bundles that make more efficient use of each transmitted data packet. Furthermore, on some complex platforms such as ~~Windows/Windows CE~~ WINDOWS™/WINDOWS CE™, a number of controls are relatively rich in features. For example, the list view controls on these operating systems allow users to change column width and scroll through the list using the scroll bars. In the preferred embodiment, the distributed UI architecture separates the UI from the data, thus allowing the client to take advantage of these features without needing assistance from the server.

Please replace the paragraph starting at page 11, line 9 with the following amended paragraph:

As used herein, a "client device" or a "presentation device" is any device or combination of devices capable of providing information to an end user of distributed UI system 100. For example, a client device 102, 104, 106 may be a personal computer, a

television monitor, an Internet-ready console, a wireless telephone, a personal digital assistant (PDA), a home appliance, a component in an automobile, a video game console, or the like. The client devices may be configured in accordance with any number of conventional platforms, while using various known operating systems (OSs). For example, the client device could be a ~~Handspring Visor~~ HANDSPRING VISOR™ running the ~~Palm OS~~ Palm OS™, a ~~Pocket PC~~ POCKET PC™ running the ~~Windows CE OS~~ WINDOWS CE OS™, a laptop computer running the ~~Windows 2000 OS~~ WINDOWS 2000 OS™, a smartphone running a custom OEM-supplied OS, or a specialized data device built with a commercially available RTOS such as Wind River's pSos. In practice, system 100 is particularly suited for use with wireless client devices, since it can handle the bandwidth limitations and inconsistent connections inherent in current wide-area wireless networks much better than existing alternatives. FIG. 1 depicts client device 104 as a wireless device or system.

Please replace the paragraph starting at page 12, line 17 with the following amended paragraph:

Communication links 112, 114 may be suitably configured in accordance with the particular communication technologies and/or data transmission protocols associated with the given client device. For example, although a communication link 112, 114 preferably utilizes broadband data transmission techniques and/or the TCP/IP suite of protocols, the link could employ NetBIOS, NetBEUI, data link control (DLC), ~~AppleTalk~~ APPLETALK™, or a combination thereof. Communication links 112, 114 may be established for continuous communication and data updating or for intermittent communication, depending upon the infrastructure.

Please replace the paragraph starting at page 12, line 25 with the following amended paragraph:

The UI server 108 preferably includes and/or communicates with one or more data sources or data servers 116, which may be configured in accordance with conventional techniques. As used herein, the data server 116 manages source data items that can be delivered to the user of the client devices. In a practical distributed UI system 100, data server 116 may manage the delivery of email, documents, PIM data, and/or any other type of data to and from the client devices. For example, the data server 116 may be realized as local, personal storage such as a ~~Microsoft Outlook~~ MICROSOFT OUTLOOK™ ".pst" file on the same computer as UI server 108, or as a ~~Microsoft Exchange Server~~ MICROSOFT EXCHANGE SERVER™, a ~~Lotus Domino Server~~ LOTUS DOMINO SERVER™, a POP3 server, an IMAP server, or the like. A given data server 116 may be integral to UI server 108, it may be a distinct component maintained at the service site associated with UI server 108, or it may be maintained by a third party unrelated to the entity responsible for maintaining UI server 108. Accordingly, data server 116 may be configured to communicate with UI server 108 over a direct communication link 118 and/or via network 110 using an indirect communication link 120.

Please replace the paragraph starting at page 13, line 9 with the following amended paragraph:

A "server" is often defined as a computing device or system configured to perform any number of functions and operations associated with the management, processing, retrieval, and/or delivery of data, particularly in a network environment. Alternatively, a "server" may refer to software that performs such processes, methods, and/or techniques. As used herein, "UI server" generally refers to a computing architecture that processes data and defines display formats for the client-side UI, while executing a number of server-based applications accessed by the client devices. As in

most commercially available general purpose servers, a practical UI server may be configured to run on any suitable operating system such as UNIX™, LINUX™, the APPLE MACINTOSH OS™, or any variant of MICROSOFT WINDOWS™, and it may employ any number of microprocessor devices, e.g., the PENTIUM™ family of processors by INTEL™ or the processor devices commercially available from ADVANCED MICRO DEVICES™, IBM™, SUN MICROSYSTEMS™, or MOTOROLA™.

Please replace the paragraph starting at page 15, line 14 with the following amended paragraph:

For the sake of illustration, the techniques of the present invention are explained herein in the context of an existing desktop email application. Of course, the distributed UI system may (and preferably does) support any number of alternate and/or additional applications. FIG. 3 is an illustration of an example UI 300 associated with a desktop email application. Although not a requirement of the present invention, the UI 300 may utilize UI components, controls, icons, and features that are utilized by standard or commercially available applications. For example, UI 300 may be an example of ~~Microsoft's Outlook, Microsoft's Outlook Express, Novell's GroupWise~~ MICROSOFT'S OUTLOOK™, MICROSOFT'S OUTLOOK EXPRESS™, NOVELL'S GROUPWISE™, or the like.

Please replace the paragraph starting at page 18, line 12 with the following amended paragraph:

Furthermore, opening large messages or attachments is much simpler because the attachment is actually being opened on the UI server, and only a single page view (and possibly some additional cached data) need be transmitted to the client at any one time. In contrast, conventional wireless PDAs (e.g., ~~Palm~~ PALM™ devices or ~~Blackberry~~ BLACKBERRY™ devices) cannot open attachments, and a wireless ~~Windows CE~~ WINDOWS CE™ device must download the entire attachment before

opening (and the user runs the risk of format loss due to document conversion, assuming the document type is supported at all). The view presented by the client device may be editable or read-only, depending upon the attachment type and/or the device capabilities.

Please replace the paragraph starting at page 23, line 28 with the following amended paragraph:

The server-based applications 710 can represent any number of different applications, features, or functions, e.g., an email application, a calendar application, an address book or contact list, a chat application, a task reminder list, an alarm feature, a messaging service, or any other application that could run on a desktop (or other) computer. These applications reside primarily at the UI server, which handles most, if not all, of the application processing on behalf of the client devices. Other than telling the client device what UI changes to make based on the current UI state and actions selected by the user, the job of the UI server is basically to be a remote data source. The primary difference between this type of data source and typical ones is simply that the client need not know the names, types, or source of the data; the UI server is responsible for obtaining and formatting the data for the client based on a data ID that the UI server associates with the control descriptions in the form definition. Notably, the UI server can be configured to communicate with and support multiple data sources for any given server-based application 710. For example, PIM applications may utilize a number of different data sources, e.g., ~~Microsoft Exchange, Starfish Organizer, Novell Communicator~~ MICROSOFT EXCHANGE™, STARFISH ORGANIZER™, NOVELL COMMUNICATOR™, and the like. Accordingly, each of the server-based applications 710 preferably contains an interface to a per-application data source module 722, which can be replaced depending on which data source is being used.

Please replace the paragraph starting at page 25, line 9 with the following amended paragraph:

The server-based applications 710 may communicate with any number of data source modules 722, which in turn obtain source data items from one or more data servers (see FIG. 1). The data source modules 722 may utilize any suitable communication protocol or model, e.g., the ~~Microsoft Outlook Object Model (OOM)~~ MICROSOFT™ OUTLOOK OBJECT MODEL (OOM), to communicate with the data servers. For example, multiple data source modules 722 may be suitably configured (in accordance with known techniques) to each communicate with one of the following server types: ~~Microsoft Exchange, Lotus Notes~~ MICROSOFT EXCHANGE™, LOTUS NOTES™, IMAP, POP3, and SMTP. Alternatively, a single data source module 722 could use a multi-source API, such as OOM, to communicate with any one of those data sources. Once obtaining the source data items, the data source modules 722 can function as an interface or an intermediary for the server-based applications 710 that process the source data items. In this respect, the server-based applications are configured to manipulate source data items for presentment and/or editing at the client device.